



ANSI/TIA/EIA-102.AAAD-2002
Approved: June 13, 2002

TIA/EIA-102.AAAD

TIA/EIA STANDARD

Project 25

Block Encryption Protocol

TIA/EIA-102.AAAD

JULY 2002

TELECOMMUNICATIONS INDUSTRY ASSOCIATION



Representing the telecommunications industry in
association with the Electronic Industries Alliance



NOTICE

TIA/EIA Engineering Standards and Publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for his particular need. Existence of such Standards and Publications shall not in any respect preclude any member or nonmember of TIA/EIA from manufacturing or selling products not conforming to such Standards and Publications, nor shall the existence of such Standards and Publications preclude their voluntary use by those other than TIA/EIA members, whether the standard is to be used either domestically or internationally.

Standards and Publications are adopted by TIA/EIA in accordance with the American National Standards Institute (ANSI) patent policy. By such action, TIA/EIA does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the Standard or Publication.

This Standard does not purport to address all safety problems associated with its use or all applicable regulatory requirements. It is the responsibility of the user of this Standard to establish appropriate safety and health practices and to determine the applicability of regulatory limitations before its use.

(From Standards Proposal Nos. 3-4921 and 3-4921-1, formulated under the cognizance of the TIA TR-8.3 Subcommittee on Encryption.)

Published by
©TELECOMMUNICATIONS INDUSTRY ASSOCIATION 2002
Standards and Technology Department
2500 Wilson Boulevard
Arlington, VA 22201 U.S.A.

**PRICE: Please refer to current Catalog of
EIA ELECTRONIC INDUSTRIES ALLIANCE STANDARDS and ENGINEERING
PUBLICATIONS or call Global Engineering Documents, USA and Canada
(1-800-854-7179) International (303-397-7956)**

All rights reserved
Printed in U.S.A.

PLEASE!
DON'T VIOLATE
THE
LAW!

This document is copyrighted by the TIA and may not be reproduced without permission.

Organizations may obtain permission to reproduce a limited number of copies through entering into a license agreement. For information, contact:

Global Engineering Documents
15 Inverness Way East
Englewood, CO 80112-5704 U.S.A. or call
U.S.A. and Canada 1-800-854-7179, International (303) 397-7956

www.docin.com

NOTICE OF DISCLAIMER AND LIMITATION OF LIABILITY

The document to which this Notice is affixed has been prepared by one or more Engineering Committees of the Telecommunications Industry Association ("TIA"). TIA is not the author of the document contents, but publishes and claims copyright to the document pursuant to licenses and permission granted by the authors of the contents.

TIA Engineering Committees are expected to conduct their affairs in accordance with the TIA Engineering Manual ("Manual"), the current and predecessor versions of which are available at http://www.tiaonline.org/standards/sfg/engineering_manual.cfm. TIA's function is to administer the process, but not the content, of document preparation in accordance with the Manual and, when appropriate, the policies and procedures of the American National Standards Institute ("ANSI").

THE USE OR PRACTICE OF CONTENTS OF THIS DOCUMENT MAY INVOLVE THE USE OF INTELLECTUAL PROPERTY RIGHTS ("IPR"), INCLUDING PENDING OR ISSUED PATENTS, OR COPYRIGHTS, OWNED BY ONE OR MORE PARTIES. TIA MAKES NO SEARCH OR INVESTIGATION FOR IPR. WHEN IPR CONSISTING OF PATENTS AND PUBLISHED PATENT APPLICATIONS ARE CLAIMED AND CALLED TO TIA'S ATTENTION, A STATEMENT FROM THE HOLDER THEREOF IS REQUESTED, ALL IN ACCORDANCE WITH THE MANUAL. TIA TAKES NO POSITION WITH REFERENCE TO, AND DISCLAIMS ANY OBLIGATION TO INVESTIGATE OR INQUIRE INTO, THE SCOPE OR VALIDITY OF ANY CLAIMS OF IPR.

ALL WARRANTIES, EXPRESS OR IMPLIED, ARE DISCLAIMED, INCLUDING WITHOUT LIMITATION, ANY AND ALL WARRANTIES CONCERNING THE ACCURACY OF THE CONTENTS, ITS FITNESS OR APPROPRIATENESS FOR A PARTICULAR PURPOSE OR USE, ITS MERCHANTABILITY AND ITS NON-INFRINGEMENT OF ANY THIRD PARTY'S INTELLECTUAL PROPERTY RIGHTS. TIA EXPRESSLY DISCLAIMS ANY AND ALL RESPONSIBILITIES FOR THE ACCURACY OF THE CONTENTS AND MAKES NO REPRESENTATIONS OR WARRANTIES REGARDING THE CONTENT'S COMPLIANCE WITH ANY APPLICABLE STATUTE, RULE OR REGULATION.

TIA SHALL NOT BE LIABLE FOR ANY AND ALL DAMAGES, DIRECT OR INDIRECT, ARISING FROM OR RELATING TO ANY USE OF THE CONTENTS CONTAINED HEREIN, INCLUDING WITHOUT LIMITATION ANY AND ALL INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS, LOSS OF PROFITS, LITIGATION, OR THE LIKE), WHETHER BASED UPON BREACH OF CONTRACT, BREACH OF WARRANTY, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY OR OTHERWISE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE FOREGOING NEGATION OF DAMAGES IS A FUNDAMENTAL ELEMENT OF THE USE OF THE CONTENTS HEREOF, AND THESE CONTENTS WOULD NOT BE PUBLISHED BY TIA WITHOUT SUCH LIMITATIONS.

Contents

1. Scope	1
1.1 Revision History	2
1.2 References	2
2. Description.....	3
3. Synchronization for Voice Messages.....	4
3.1 Initialization Vector.....	6
4. Keystream Generator	6
4.1 Output FeedBack (OFB) Description	7
4.2 Encryption Input Register Initialization	8
4.3 Block Encryption System	11
4.4 Key Selection	13
5. Voice Operation.....	14
5.1 Unencrypted Operation.....	14
5.2 Clock Schedule	14
5.3 Encryption Bit Order.....	16
6. Data Operation	20
6.1 Encrypted Data Packet Structure.....	21
7. Mandatory Algorithm	24
Annex A (normative) – Data Encryption Standard (DES)	25
A.1 Algorithm ID and Description	25
A.2 Output FeedBack (OFB) Description	25
A.3 Bit Correspondence	25
A.4 Voice Encryption Example.....	26
Annex B (normative) – Triple Data Encryption Algorithm (TDEA),.....	29
B.1 Algorithm ID and Description	29
B.2 Output FeedBack (OFB) Description	29
B.3 Bit Correspondence	29
B.4 Voice Encryption Examples	30
Annex C (normative) – Advanced Encryption Standard (AES).....	33
C.1 Algorithm ID and Description.....	33
C.2 Output Feedback (OFB) Description	33
C.3 Bit Correspondence.....	33
C.4 Voice Encryption Example	34

Terminology

AES	Advanced Encryption Standard – a block encryption algorithm with a 128-bit input register (n) and a key variable length of 128, 192, or 256 bits (k)
ALGID	Algorithm ID to indicate the type of encryption algorithm
ARQ	Automatic Retry Request to retry corrupted data packets
BR	Base Radio, a reference designating a base station
CAI	Common Air Interface, described in Ref. 3
Cipher Text	Encrypted information, sometimes called 'coded,' the result of Plaintext exclusive ORed with Keystream
CON	Console, a standard reference model designation
CRC	Cyclic Redundancy Checksum for data error detection
DES	Data Encryption Standard – a block encryption algorithm with a 64-bit input/output register (n) and a key variable length of 64-bits (k, but every eighth bit going from left to right is a parity bit)
ES	Encryption Synchronization information embedded in voice
IMBE	Improved Multi-Band Excitation coder for voice
IV	Initialization Vector, the starting point of the encryption algorithm for each transmission
Key Variable	Affects how the algorithm converts plain text to cipher text
Keystream	The output of the block encryption algorithm that is exclusive ORed with Plain Text to form Cipher Text
KID	Key Identifier to indicate the encryption key for the message
LDU	Logical Link Data Unit, one of two data units that comprise a voice superframe
LFSR	Linear Feedback Shift Register
LLC	Logical Link Control sublayer of the OSI Data Link Layer
LSD	Low Speed Data embedded in voice
MI	Message Indicator, used to synchronize encryption
MR	Mobile Radio, a reference designating a mobile or portable radio
Octet	8 bits grouped together, also called a byte
OFB	Output Feed Back, one operating mode for block encryption
OSI	Open System Interconnection reference model
Plain Text	Unencrypted information, sometimes called 'clear'
RFG	RF system Gateway, a standard reference model designation
SAP	Service Access Point, where a network provides a service
TDEA	Triple Data Encryption Algorithm – a block encryption algorithm with a 64-bit input register and a key variable length of 64, 128, or 192 bits (k, but every eighth bit going from left to right is a parity bit). Also known as Triple DES.

Foreword:

(This foreword is not part of this document)

This Standard was developed with inputs from the TIA-APCO Project 25 Interface Committee (APIC), the APIC Encryption Task Group and TIA Industry Members through the TR-8.3 Encryption Subcommittee.

This Standard will be maintained by the TR-8.3 Subcommittee under the sponsorship of TIA.

This Standard describes the encryption protocol for land mobile radios meeting the Project 25 requirements. For system implementations, such as Trunking, Data, etc. this document provides only the appropriate requirements to ensure over-the-air encryption compatibility, rather than all the requisite specification details for such implementations. This document will be updated as necessary to ensure a compatible encryption protocol as additional system implementation requirements are available.

This Standard expands the material given in TIA/EIA-102.AAAA, *Project 25 DES Encryption Protocol*, February 2001. However, this standard incorporates and is completely compatible with that standard.

For information on specific implementations, as they are developed, the reader is referred to the Project 25 System and Standards Definition document (TIA/EIA TSB102-A) for; a Project 25 Overview, a General System Model, other applicable standards, a Glossary, and a Statement of Requirements.

This document includes 3 normative annexes.

www.docin.com

Patent Identification

The reader's attention is called to the possibility that compliance with this document may require the use of one or more inventions covered by patent rights.

By publication of this document no position is taken with respect to the validity of those claims or any patent rights in connection therewith. The patent holders so far identified have, we believe, filed with APCO/NASTD/FED statements of willingness to grant licenses under those rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such licenses. Details may be obtained from APCO/NASTD/FED.

The following patent holders and patents have been identified in accordance with the TIA intellectual property rights policy: none.

TIA shall not be responsible for identifying patents for which licenses may be required by this document or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

1. SCOPE

The Project 25 standard covers all of the parts of a system for public-safety Land Mobile Radio communications. These systems include portable radios for hand held operation, mobile radios for vehicular operation, base stations for fixed installations, and other fixed equipment for wide area operation and console operator positions, as well as computer equipment for data communications. The standard defines the means for this equipment to send and receive digital information, in the form of either voice or data (i.e. non-voice) messages.

The reader of this document should be familiar with Appendix C, STATEMENT OF REQUIREMENTS, of reference 1, *Project 25 Systems and Standards Definition*. One requirement is to be able to protect digital communications with encryption. The implication is that a means for encryption and decryption must be included in those system elements (e.g. portable and mobile radios) that intend to protect the messages they exchange in conformance to the other parts of Project 25 standard. It is the scope of this document to describe how to use encryption to protect messages for the Project 25 standard.

Reference 1 also contains a General System Model to describe a Project 25 standard system. Such a system is decomposed into functional groups with simple designations like 'MR' for mobile radio, and 'BR' for base station (radio). These functional groups are loosely correlated to real products and devices, but they need not be. Each functional group has one or more interfaces to other functional groups that allow information to be transferred through the system. The interface for communications over a radio channel is called the Common Air Interface (CAI). The formats for transmission of information over the Common Air Interface are described in reference 3, *Project 25 FDMA CAI*.

The functions of encryption and decryption generally take place near the end points of a message path in a system, in order to maintain the confidentiality of the information through as much of the system as possible. This means that the encryption and decryption functions can be provided at points where voice information is coded with IMBE, such as an MR (mobile or portable radio) or a CON (console). The functions may also be provided at points where data (non-voice) information enters the system such as an RFG (RF system Gateway). This is diagrammed in figure 1-1 for clarity.

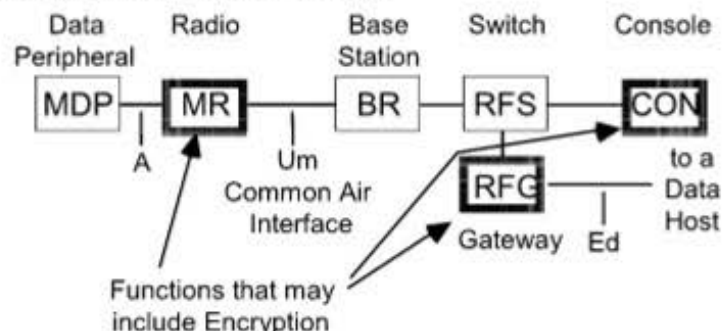


Figure 1-1 Subset of Project 25 General System Model

This Block Encryption Protocol defines the operation of encryption and decryption in a way that is compatible with information transfer through a Project 25 standard system, and especially, through the CAI of such a system. It is important to note that the Block Encryption Protocol is not actually part of the CAI because encryption and decryption may take place in system devices that are not directly connected to the CAI.

1.1 Revision History

Version 0.0, 7Jun2000, Draft released.

Version 0.1, 2Aug2000, Draft 2 with general cleanup, added annexes for DES, Triple DES, and AES.

Version 0.2, 27Sep00, Draft 3 with general cleanup, added Plain Text and Cipher Text to annexes, added actual values for annexes A and B, moved references to *Project 25 DES Encryption Protocol* to Foreword.

SP-3-4921, 28 March 2001, prepared document for ballot (same content as draft 3 with clarified DES interoperability in Annex B).

SP-3-4921-1, 14 January 2002, prepared document for default ballot for technical changes made to section 6, and Annex C as a result of comment resolution.

TIA/EIA-102.AAAD, 3 June 2002, minor editorial corrections from default ballot.

1.2 References

The following standards contain provisions that, through reference in this text, constitute provisions of this Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. ANSI and TIA maintain registers of currently valid national standards published by them.

1. TIA/EIA TSB-102-A, *APCO Project 25 System and Standards Definition*, November 1995.
2. TIA/EIA-102.BABA, *Project 25 Vocoder Description*, May 1998.
3. TIA/EIA-102.BAAA, *Project 25 FDMA Common Air Interface*, May 1998,
4. TIA/EIA-102.BAAC, *Project 25 Common Air Interface Reserved Values*, May 2000,
5. NIST, *DES Modes of Operation*, FIPS Publication 81.

2. DESCRIPTION

In the GENERAL SYSTEM MODEL section in reference 1, *Project 25 System and Standards Definition*, the interface specifications are organized with the Open System Interconnection (OSI) reference model into seven layers. In the case of the encryption function, there is no single system interface that is used. Instead, there is an encryption protocol that exists across several interfaces to allow the confidential exchange of information through the system. This protocol exists above the layer 2 functions defined in reference 3, *Project 25 FDMA CAI*. It may be considered to be part of an upper sublayer of Layer 2, or part of Layer 3 or Layer 4. This is shown in fig. 2.1.

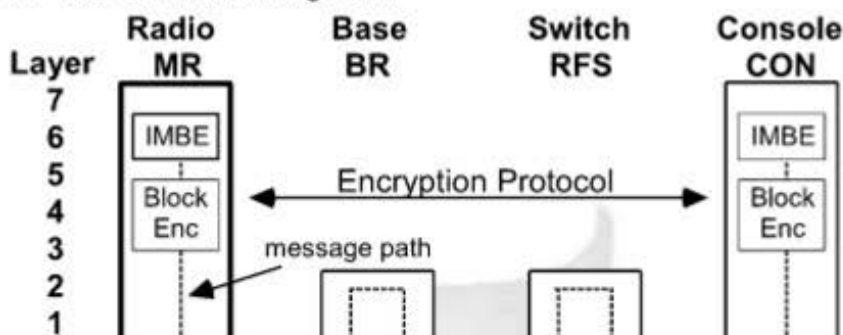


Figure 2-1 Typical Protocol Context for Voice

This document describes the following items that are necessary for the encryption protocol:

- Encryption Algorithm with Key Variable -- obviously required
- Operating Mode -- specifies how the algorithm is used
- Initialization Vector -- unpredictably initializes the algorithm
- Message Indicator -- necessary to synchronize encryption

The protocol defined here is compatible with either voice or data messages and can be transported through a radio network using the CAI.

The interconnection between these components is shown in fig. 2-2. The encryption algorithm is abstractly embedded within a synchronization function. The data to be encrypted is referred to as plain text and the encrypted data is referred to as cipher text.

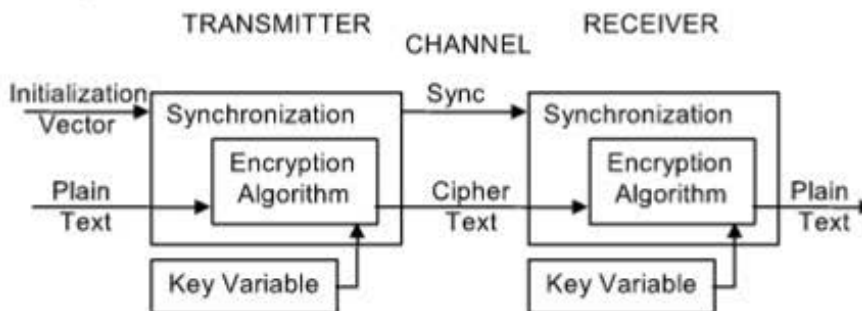


Figure 2-2 Components of Encryption

3. SYNCHRONIZATION FOR VOICE MESSAGES

To be able to decrypt messages, the receiver decryptor must be in the same state as the transmitter encryptor. The CAI provides space for up to 72 bits of this synchronization information in the Message Indicator (MI) vector at the beginning of the message (in the header), and periodically during the message in the LDU2 portion of the voice superframe. The information in the MI is used to set the state of both the transmitter and receiver. As long as the MI information is received correctly, the receiver and transmitter are synchronized. The periodic MI allows a receiver to begin decrypting the message in the middle, which is useful for voice messages when the beginning of the message has been lost.

The MI values could be randomly determined each time they are sent. However, by having a rule which allows subsequent MIs to be computed from a previous MI the receiver can determine whether it is still in synchronization with the transmitter by comparing its predicted version of the MI with the transmitter's MI.

The rule used for this protocol is a Linear Feedback Shift Register (LFSR) counter. The LFSR uses a 64-bit register that may be conveniently implemented in either a hardware or a software design. The characteristic polynomial used for the LFSR is

$$C(x) = 1 + x^{15} + x^{27} + x^{38} + x^{46} + x^{62} + x^{64}.$$

A simple circuit for generating the LFSR sequence is diagrammed below. The circuit consists of a 64-stage shift register that shifts from right to left. The outputs of stages 15, 27, 38, 46, 62, and 64 (the last stage) are added together, modulo 2, to create the feedback input to the first stage. Given that the current MI is in the register, the next MI shall be the register contents after 64 shifts.

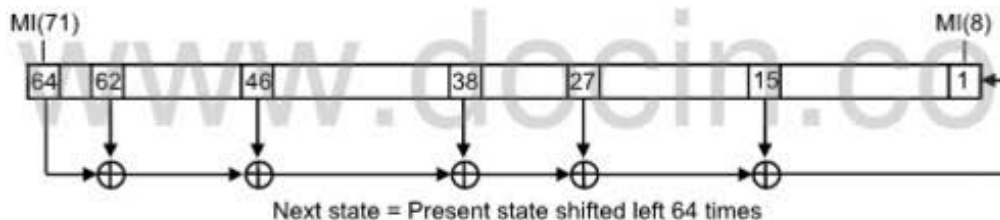


Figure 3-1 LFSR MI Generator

The choice of a 64-bit register is largely determined by the requirement to have a very long sequence in order to maintain secure encryption. This sequence is a maximal length sequence so its repeat period is $2^{64}-1$ clock cycles. In order for the receiver to be synchronized to the transmitter, it is necessary for 64 bits to be transferred in an error-free manner from the transmitter to the receiver. The maximum number of bits that the CAI allots to MI is 72 bits. The only consequence of using 64 bits rather than 72 is that the repeat period is shortened by a factor of about 256. In general, this reduction can be minimized by changing the key variable more frequently.

The 72 MI bits in the CAI are numbered from 0 to 71 with MI(71) being transmitted first. Bits MI(0) through MI(7) are not used for this block encryption

protocol, so they are reserved and set to nulls. Bits MI(8) through MI(71) map to bits 1 through 64, respectively, of the LFSR shown in figure 3-1.

The LFSR process to compute a new MI is done once for each superframe and is sent in the header and each even LDU, in the case of voice messages. For the next superframe, another MI is computed with the LFSR process given above. As an example, the table below shows a typical LFSR calculation.

Table 3-1 LFSR MI Generator Example

		MI vectors					
Header		1234	5678	90AB	CDEF	00	
Superframe #	1	6FE2	802A	A403	828B	00	
	2	FF35	B190	9449	B835	00	
	3	7EAD	FBE0	D23E	CDDF	00	
	4	34B6	7525	ADEE	0EC7	00	
	5	0401	7ED2	40D1	7BCF	00	
	6	4BFF	2D26	B42E	8B31	00	
	7	D9FB	26E7	FD9F	0B40	00	
	8	1F1F	6114	075B	0B8F	00	
	9	4B50	734F	85B6	2BA2	00	
	10	74B8	ED38	F959	E1D0	00	
	11	19C1	D2BB	4655	DD1E	00	
	12	A9B4	26C8	2ACA	53BE	00	
	13	4AD9	DDF5	D6E1	422F	00	
	14	9945	03D6	14B3	682E	00	
	15	ED6B	4CE8	69C9	9E95	00	
	16	CED8	B11D	5AAF	2B91	00	
	17	4662	5D96	AC29	FFAD	00	
	18	0964	DBFD	CD7F	52A2	00	
	19	BE9A	55DE	4004	B0D7	00	
	20	0BB2	1DF3	92C0	0EC6	00	
	21	D973	4B3B	41A1	7916	00	
	22	BB57	B0CF	F2FA	D2DA	00	
	23	5013	A83D	4937	029A	00	
	24	6658	6FE2	1AD2	39CA	00	
	25	E108	0889	8D27	719C	00	
	26	26EC	FDA9	299C	79B4	00	
	27	2FBA	B56A	2401	2E1D	00	
	28	3077	5736	D837	DC5C	00	
	29	C853	AF7D	EE34	F108	00	

with the following correspondence of hex to MI bit values:

Hex	1	2	3	4	5	...	D	E	F	0	0	
Binary	00010010001101000101	...	11011110111100000000									
MI#	7	6	6	6	5	5	...	1	1	8	4	0
	1	8	4	0	6	2		6	2			

3.1 Initialization Vector

The first MI in the header is known as the Initialization Vector (IV). It sets the sequence of the MIs to follow using the LFSR sequence generator. For the best security, the MIs with a given key variable should not be reused. Since the MI initializes the encryption algorithm, it also determines the keystream that is produced by the encryption algorithm with a key variable. Reusing an MI would result in the same keystream being produced. If two messages use the same keystream, then exclusive ORing them together removes the keystream and leaves just the messages exclusive ORed together. From this, it is possible to recover information about the messages, such as if parts of the message are the same. Having different MIs for each message greatly increases the task of cryptanalysis.

The LFSR sequence used to generate the MIs eventually repeats. In this sense we really have a very large 'ring' of MI values. Since each MI results in a specific piece of keystream, the MI ring is converted to a 'keystream ring' by the encryption and key variable. For a specific block encryption algorithm there are 2^k keystream rings from a given MI ring.

Since this protocol only addresses compatibility, it does not specify the means of creating the IV. The protocol will work if the same IV is used for each message, although with very little security. Manufacturers should consider methods to assure that the IV is different for each message. One method would generate a random IV at each transmission, in essence hopping around the keystream ring. Probability calculations could be used to determine the time interval of a key variable change based on the number of messages and their length. Another method would give each radio a portion of the keystream ring. For each transmission, the radio would step along this piece of the ring. Once the allotted portion is used, a warning could be given that a key variable change is recommended.

If the MI LFSR generator is loaded with 64 zeroes it will generate all zeroes as the MI for the rest of the message. This would result in the keystream being the same for each superframe of the message resulting in no security. Consequently, an IV value of all zeroes should be considered invalid. In fact, an all zeroes MI has been reserved (see reference 4, *Project 25 CAI Reserved Values*) as one of the indicators of an unencrypted message. If the procedure for generating the IV produces this null value, it should repeat the generation procedure until it generates a valid IV.

4. KEYSTREAM GENERATOR

The Keystream Generator defined in this protocol uses an n-bit block encryption algorithm in the Output FeedBack (OFB) mode. The output of the Keystream Generator is exclusive ORed with Plain Text to produce Cipher Text. In general, the Keystream Generator requires n bits for initialization and for each pass through the block encryption algorithm produces n bits of keystream. However,

this protocol shall limit the number of keystream bits to be the leftmost bits that are a multiple of 8. Any remaining rightmost keystream bits shall be discarded.

An n -bit block encryption algorithm operates on n -bit input vectors with a k -bit key variable to produce encrypted n -bit output vectors. The encryption process consists of permutations, and non-linear substitutions usually done in multiple rounds and is controlled by the key variable. The Data Encryption Standard (DES) is an example of a block algorithm with $n = 64$ and $k = 56$. The remainder of this document will have blocks that refer to n -bit input and output vectors and will show the entire block encryption operation as a block labeled 'bEnc.'

4.1 Output FeedBack (OFB) Description

OFB takes the contents of the encryption output register and places it back in the input register for the next iteration of the encryption operation (for a description of OFB for the DES block algorithm, see reference 5, *DES Modes of Operation*). All n bits of the output register are placed into the input register in the same order, the leftmost bit of the output register going to the leftmost bit of the input register, the rightmost bit of the output register going to the rightmost bit of the input register, etc. This document will henceforth refer to this bit ordering as left-to-left and right-to-right. Each iteration of the block algorithm will yield an n -bit block in the output register for encryption in the transmitter or decryption in the receiver. This is shown in figure 4-1

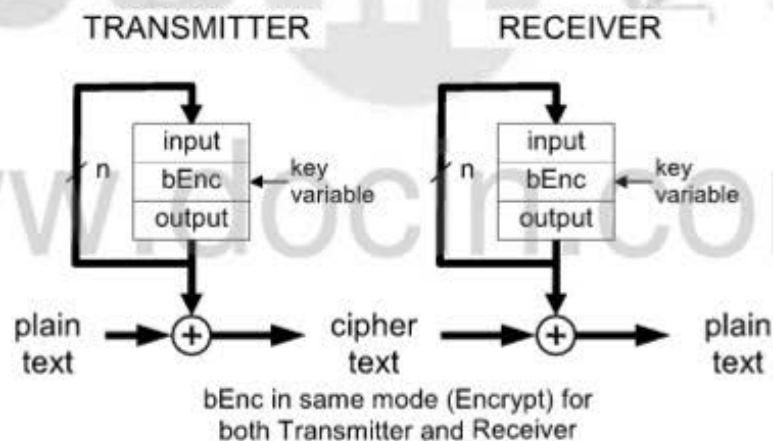


Figure 4-1 OFB Operation

Block algorithms are reversible, which is to say they can encrypt the input register contents to yield the output register contents, or they can decrypt the output register contents to yield the input register contents. For this protocol, OFB operation shall always use the encrypt mode for both the transmitter and receiver.

In this protocol, the initialization is derived from the MI (as described in the next section, 4-2). The keystream is derived by iterating the computation of the OFB operation. To generate enough bits to encrypt the information in a superframe,

multiple OFB iterations are required. To provide a greater degree of security, the first iteration, which encrypts the Message Indicator, is not used for encrypting plain text. Also, as mentioned above, only multiples of 8 bits (the leftmost) are used as the keystream. Any remaining bits (the rightmost) are not used for encryption, but they are fed back to the input register. OFB is chosen as the mode of operation for encryption because it has the property of giving only one error for each error made on the channel.

The following parameters are useful to describe the general operation of the Block Encryption Protocol:

- n = number of bits in an encryption algorithm block
- r = number of bits left after dividing the encryption block into octets
(this number of bits is discarded from the rightmost bits of the block encryption output register for encryption of Plain Text)
= (n) modulo 8
- m = number of octets in an encryption block
= integer part of $(n/8) = (n - r)/8$
- B = total number of OFB iterations to encrypt one superframe
= $1 + [(213/m) \text{ rounded up to an integer}]$
- # = Plain Text octet number (see section 5.3 for correspondence with CAI)
- B# = encryption block number
= output register after B# OFB encryption iterations
= $1 + [(#/m) \text{ rounded up to an integer}]$
- m# = octet number in an encryption block
= $(# - 1) \text{ modulo } m$

4.2 Encryption Input Register Initialization

Since the size of the input register in the block algorithm is n bits, and the MI vector size is 64 bits a consistent method of filling the input register must be given for the case where n does not equal 64.

For $n \leq 64$, the leftmost bits of the LFSR register are loaded into the encryption input register, left-to-left and right-to-right. While this protocol defines what is to be done for $n < 64$, it is unlikely that a block encryption with this condition will ever be used. Manufacturers should keep in mind that there will be more conditions placed on the IV when $n < 64$ to keep the keystream from repeating.

For $n > 64$, a rule is required for filling the bits in excess of the 64 bits provided by the MI. There are many possibilities for this rule, such as filling unused bits with a fixed value, and most are cryptographically sound. However, there may be some flaw in the encryption algorithm that can be exploited by using a fixed value. To get around this, an expansion of the 64 MI bits is done that varies the remaining bits based on the given 64 bits.

The LFSR that is used to generate MIs can be used for this function. Recall from section 3, that the LFSR sequence is really a ring of MI values. By concatenating successive MIs, the MI field can be expanded to larger values. For this protocol, this is done by using an n -stage shift register with the LFSR rule placed on the rightmost 64 bits of this register. This is shown in figure 4-2. The 64 leftmost bits of the MI [(71)-(8)] are loaded into the shift register, it is shifted $n - 64$ times to give an n bit vector to load the encryption input register. Note that the 64 bits of the MI shall always be the leftmost bits of the expanded register after shifting $n - 64$ times.

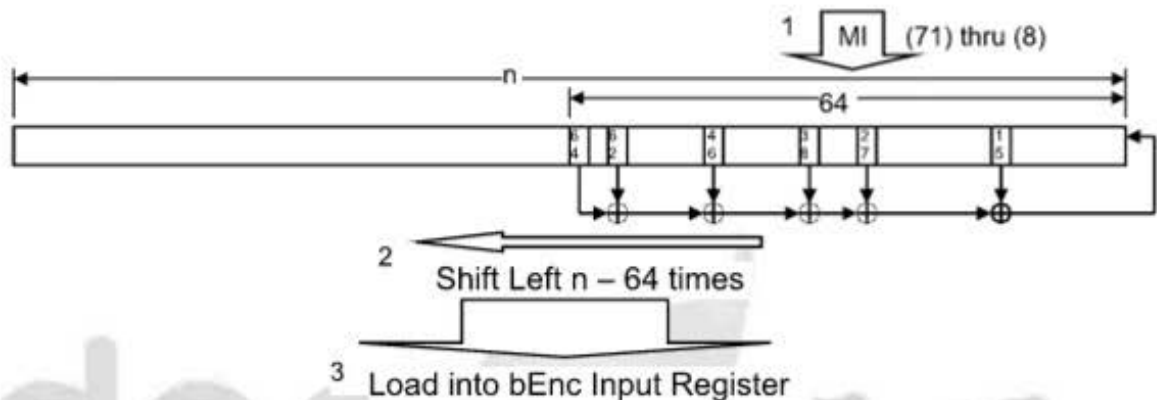


Figure 4-2 MI Expansion

Table 4-1 shows a sample calculation of this for $n = 128$ after step 3 in figure 4-2 for the MI values shown in table 3-1. However, this table also illustrates all values of $n \leq 128$. The encryption input register fill will always be the n leftmost bits of this example. The bits are loaded left-to-left and right-to-right of the expansion register to the encryption input register.

Table 4-1 MI Expansion Example

Header	1234	5678	90AB	CDEF	6FE2	802A	A403	828B
1	6FE2	802A	A403	828B	FF35	B190	9449	B835
2	FF35	B190	9449	B835	7EAD	FBE0	D23E	CDDF
3	7EAD	FBE0	D23E	CDDF	34B6	7525	ADEE	0EC7
4	34B6	7525	ADEE	0EC7	0401	7ED2	40D1	7BCF
5	0401	7ED2	40D1	7BCF	4BFF	2D26	B42E	8B31
6	4BFF	2D26	B42E	8B31	D9FB	26E7	FD9F	0B40
7	D9FB	26E7	FD9F	0B40	1F1F	6114	075B	0B8F
8	1F1F	6114	075B	0B8F	4B50	734F	85B6	2BA2
9	4B50	734F	85B6	2BA2	74B8	ED38	F959	E1D0
10	74B8	ED38	F959	E1D0	19C1	D2BB	4655	DD1E
11	19C1	D2BB	4655	DD1E	A9B4	26C8	2ACA	53BE
12	A9B4	26C8	2ACA	53BE	4AD9	DDF5	D6E1	422F
13	4AD9	DDF5	D6E1	422F	9945	03D6	14B3	682E
14	9945	03D6	14B3	682E	ED6B	4CE8	69C9	9E95
15	ED6B	4CE8	69C9	9E95	CED8	B11D	5AAF	2B91
16	CED8	B11D	5AAF	2B91	4662	5D96	AC29	FFAD
17	4662	5D96	AC29	FFAD	0964	DBFD	CD7F	52A2
18	0964	DBFD	CD7F	52A2	BE9A	55DE	4004	B0D7
19	BE9A	55DE	4004	B0D7	0BB2	1DF3	92C0	0EC6
20	0BB2	1DF3	92C0	0EC6	D973	4B3B	41A1	7916
21	D973	4B3B	41A1	7916	BB57	B0CF	F2FA	D2DA
22	BB57	B0CF	F2FA	D2DA	5013	A83D	4937	029A
23	5013	A83D	4937	029A	6658	6FE2	1AD2	39CA
24	6658	6FE2	1AD2	39CA	E108	0889	8D27	719C
25	E108	0889	8D27	719C	26EC	FDA9	299C	79B4
26	26EC	FDA9	299C	79B4	2FBA	B56A	2401	2E1D
27	2FBA	B56A	2401	2E1D	3077	5736	D837	DC5C
28	3077	5736	D837	DC5C	C853	AF7D	EE34	F108
29	C853	AF7D	EE34	F108	87DD	5737	E51F	73C1

Diagram illustrating bit lengths for expansion:

- $n=64$ (bits 1-16)
- $n=88$ (bits 1-24)
- $n=128$ (bits 1-28)

4.3 Block Encryption System

Figure 4.-3 shows the entire block encryption system

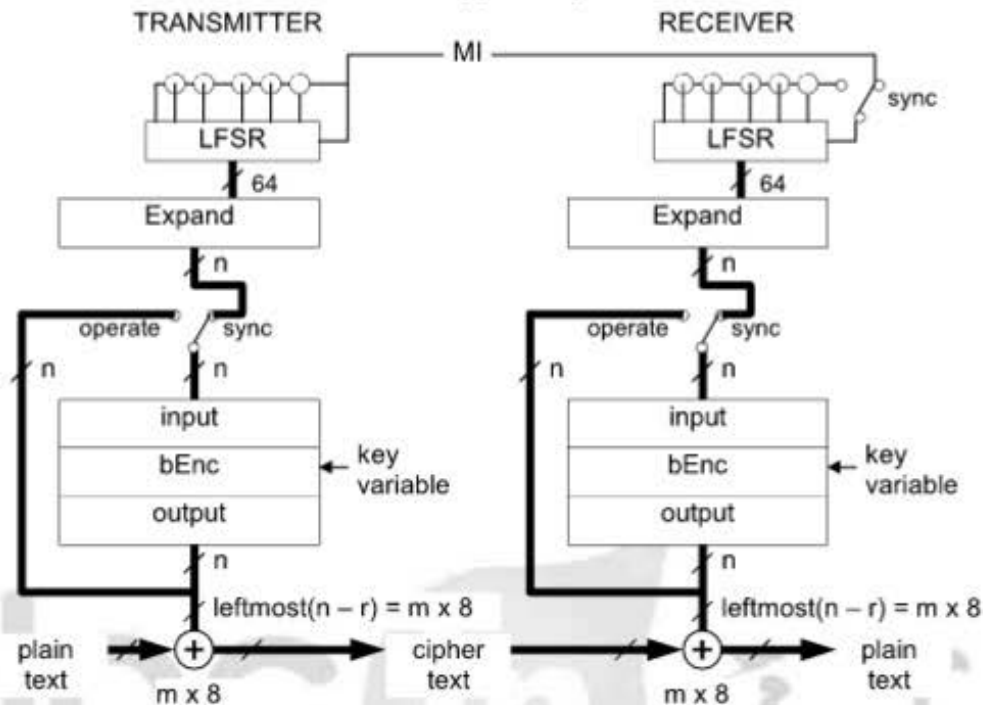


Figure 4-3 OFB Operation with LFSR Generated MI

With this diagram in mind, the transmitter operates with the following steps:

- Step 1. Key Variable Load. The Key Variable for the message is selected and loaded into the algorithm.
- Step 2. LFSR Initialization. The LFSR register is initialized with 64 bits from the initialization vector. See section 3.1 for details. These 64 bits are also transmitted over the CAI in the Header MI field.
- Step 3. Keystream Generator Initialization. If $n \leq 64$, the leftmost bits of the LFSR are loaded into the encryption input register (left to left, right to right). If $n > 64$, the LFSR register is expanded as described in 4.1 and loaded into the input register (left to left, right to right).
- Step 4. Generate MI. The LFSR register is shifted 64 times to produce the next MI value. This value is transferred to the MI register to be transmitted in LDU2 of this superframe.
- Step 5. Encrypt Superframe using OFB Operation.
 - Step 5.1 One OFB iteration is done, but the m-octet result is not used for encryption. This step initializes the input register with the encrypted MI.

The superframe plain text is separated into blocks of m octets each. Each block is encrypted by the following steps.

 - Step 5.2 Algorithm Iteration. The encryption algorithm is iterated once. This generates an n bit result.

- Step 5.3 Encryption. The leftmost m octets of the output value are exclusive ORed with the plain text to yield the cipher text. The cipher text is transmitted over the channel to the receiver.
- Step 5.4 Output Feed Back. All n bits of the output are fed back to the input register, leftmost output to leftmost input continuing on to rightmost output to rightmost input.
- Steps 5.2 through 5.4 are repeated until all encryptable bits in the superframe have been processed.
- Step 6. Repeat for Rest of Message. Go back to Step 3 until all the message has been encrypted.



4.4 Key Selection

Multiple encryption keys can be stored in radio equipment conforming to the standard. In order to identify the keys, they are stored with an associated label called a Key Identifier or KID. The type of algorithm to be used with the key is identified by an Algorithm ID or ALGID. These storage elements are shown in fig. 4-6.

Key ID (16 bits)	Key Variable	Algorithm ID (8 bits)
	traffic key 1	
	traffic key 2	
...	...	...
	traffic key n	

IV (64 bits)

Figure 4-6 Storage for Multi-Key Equipment

The transmitter may select a key in several ways. The key may be selected (a) by a manual selection of the operator, (b) by a default assignment programmed into the radio, or (c) under the influence of received traffic. For example, with (c), a recently received message may have selected a key for decryption, and that key might be used for subsequent transmissions. Once a key has been selected for a transmission, the associated KID and ALGID are embedded in the message.

The receiver examines both the KID and the ALGID to determine if it possesses the correct encryption key variable. If so, it can then decrypt the message.

The ALGID has a reserved value for unencrypted messages and also reserved values for block encryption algorithms that are allowed in the Project 25 system. These reserved values are given in reference 4, *CAI Reserved Values*.

The KID has a reserved value of \$0000. This value is used as a default KID by equipment which does not operate in a multi-key system. This value means that a default key variable is to be used on the channel. Normally, all of the equipment on such a channel will store only a single key variable and use the default KID. The single key equipment will then ignore any messages which originate from multi-key equipment using non-default key variables, because those messages should not use the default KID. The default KID value is also used for unencrypted traffic, which is signified by the reserved ALGID value.

5. VOICE OPERATION

The general description of the encryption protocol is given in section 4. Specific details for voice operation are given here. The specific details include the following items.

Unencrypted Operation. This defines the default values for the MI, KID, and ALGID for unencrypted operation.

Clock Schedule. This defines the number of times that LFSR is clocked, as well as when and how it is clocked. This also defines the number of times that the OFB operating mode is iterated to encrypt each superframe of voice.

Order of Bits. This defines the order in which the bits are encrypted. It also defines how they are combined with the n-bit output vectors of each OFB iteration.

5.1 Unencrypted Operation

When a voice message is unencrypted, the values for the MI, KID, and ALGID are shown in table 5-1.

Table 5-1 Unencrypted Default Values in Hexadecimal

Name	Value	Size
MI	\$00000000000000000000	72 bits
KID	\$0000	16 bits
ALGID	\$80	8 bits

5.2 Clock Schedule

For encrypted messages, the block encryption algorithm is operated in the n-bit OFB mode as described in section 4. It is periodically initialized with the contents of a 64-bit LFSR, also as described in section 3.

At the beginning of the transmission, the IV is loaded into the LFSR, expanded (if needed) and loaded into the block encryption input register, and placed into the MI field of the Header appended with 8 zeroes. This is depicted in fig. 5-1 as the MI in the Header.

In the CAI, each superframe has 213 octets of information that may be encrypted [consisting of 18 frames of IMBE voice information with 11 octets per frame, 4 octets of Low Speed Data (LSD), 8 octets of Link Control information (LC), and 3 reserved octets]. Since each OFB iteration encrypts m octets, and the first OFB iteration is not used, $B = 1 + [(213/m) \text{ rounded up to an integer}]$ OFB iterations are needed to encrypt the plain text in each superframe.

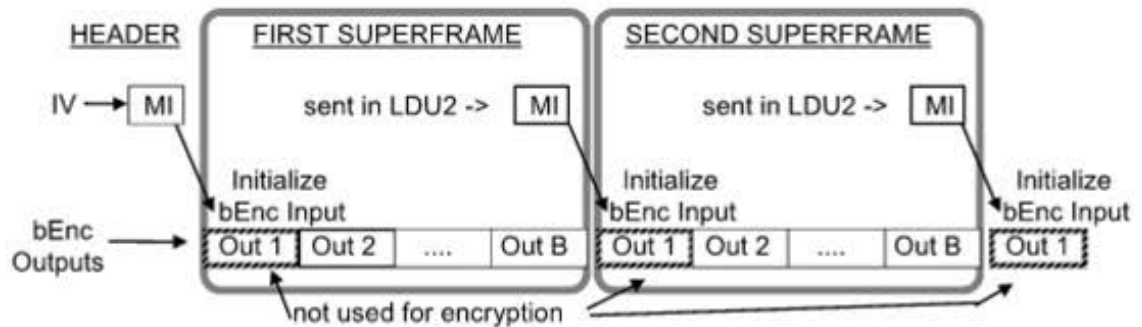


Figure 5-1 LFSR and OFB Clocking

After this MI has been generated, it is used to encrypt the contents of the first superframe. This is done by iterating the block encryption algorithm in the OFB mode B times to generate B blocks of m octets each. This generates $B \times m$ octets, of which only 213 octets are used for information. The first m octets in block 1 (Out 1) and last $[m \times (B - 1) - 213]$ octets in block B (Out B) are not used in the transmitter for encrypting plain text.

The first superframe contains two Logical Link Data Units or LDU's. Another MI value is sent in the second LDU, called LDU2. This MI is used in the receiver to initialize the decryption operation for the next superframe. To compute this value, the LFSR is clocked 64 times to get the next state and this is placed in the MI field of LDU2 appended with 8 zeroes.

After the B -th iteration of the OFB mode of operation, the transmitter is ready to begin encrypting the second superframe. The LFSR register (already transmitted in LDU2) is expanded (if needed) and loaded into the block encryption input register.

After the input register has been initialized by the expanded LFSR, the block encryption algorithm is operated B times in the OFB mode, as described before. The encryption operation in the second superframe is the same as in the first superframe, as already described.

5.3 Encryption Bit Order

The voice information within a superframe includes the components tabulated below. Table 5-2 gives the components in the order in which they are encrypted.

Table 5-2 Superframe Encrypted Information

Name	Size (octets)
Reserved	3
LC_information	8
IMBE frame 1	11
...	...
IMBE frame 8	11
Low Speed Data	2
IMBE frame 9	11
...	...
IMBE frame 17	11
Low Speed Data	2
IMBE frame 18	11
total	213

The LC information that is encrypted consists of the right most, or least significant, 64 bits of the Link Control field, as shown in table 5-3. This may include various fields of identification information, as defined in reference 3 *Project 25 FDMA CAI*.

Table 5-3 Link Control Information Octets

Name	Information (MSB, ... LSB)
L1	LC_information(63), LC_information(62), ... LC_information(56)
L2	LC_information(55), LC_information(54), ... LC_information(48)
L3	LC_information(47), LC_information(46), ... LC_information(40)
L4	LC_information(39), LC_information(38), ... LC_information(32)
L5	LC_information(31), LC_information(30), ... LC_information(24)
L6	LC_information(23), LC_information(22), ... LC_information(16)
L7	LC_information(15), LC_information(14), ... LC_information(8)
L8	LC_information(7), LC_information(6), ... LC_information(0)

The IMBE information is grouped into octets for encryption. The basic information output of the voice coder consists of a series of words designated u_0 , u_1 , ... u_6 , and u_7 . The details of the composition of the IMBE words is given in reference 2, *Project 25 Vocoder Description*. These words are grouped into 11 octets designated w_0 , w_1 , ... w_9 , w_{10} , as shown in table 5-4. The bits in the words are numbered from right to left, with the right most bit being numbered 0 and corresponding to the least significant bit. The words are encrypted by a bit-wise exclusive-or between the octet of voice information and a corresponding octet of the block encryption output register contents. For example, a Plain Text octet of \$A5 exclusive ORed with an Output Register octet of \$E8 would give \$4D as the Cipher Text.

Table 5-4 IMBE Information Octets	
Name	MSB Information LSB
w0	u ₀ bits 11 .. 4
w1	u ₀ bits 3 .. 0 u ₁ bits 11 .. 8
w2	u ₁ bits 7 .. 0
w3	u ₂ bits 11 .. 4
w4	u ₂ bits 3 .. 0 u ₃ bits 11 .. 8
w5	u ₃ bits 7 .. 0
w6	u ₄ bits 10 .. 3
w7	u ₄ bits 2,1,0 u ₅ bits 10 .. 6
w8	u ₅ bits 5 .. 0 u ₆ (10, 9)
w9	u ₆ bits 8 .. 1
w10	u ₆ (0) u ₇ bits 6 .. 0
	7 6 5 4 3 2 1 0

The Low Speed Data information consists of 4 octets. The first 2 octets are sent in LDU 1 and the last 2 octets are sent in LDU 2. They are sent consecutively, with each octet protected by a (16,8,5) cyclic code as described in Ref. 3, *Project 25 FDMA Common Air Interface*. The first octet to be sent is designated LSD_info_1. The second octet is designated LSD_info_2, etc.

The Block Encryption Output Register is depicted in figure 5-2. The register is n bits wide. A standard numbering scheme for bits numbers each bit, n#, from 0 to n-1 going from right to left. Whatever numbering scheme is used in the block encryption algorithm, this protocol considers the leftmost bit to be n-1 and the rightmost bit to be 0. It is common for block encryption to have n as a multiple of 8 [r = (n) modulo 8 = 0]. This is the case shown in Figure 5-2. If n is not a multiple of 8, then the rightmost bits shall be dropped (bits 0 ... r-1) and not used for encryption. The octets of the output register are arbitrarily given numbers from 0 to m-1 going from left to right.

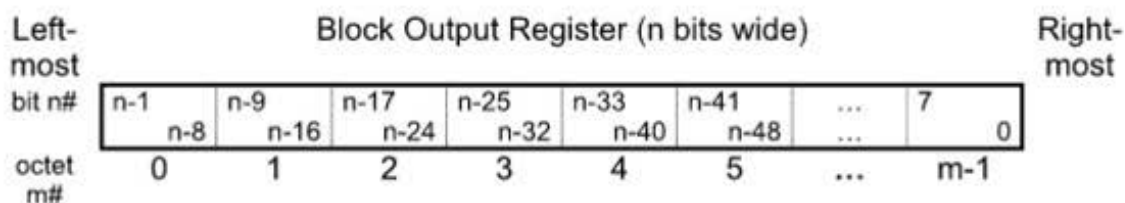


Figure 5-2 Block Encryption Output Register Octets

The Output register octets are combined with the plain text information octets to yield the cipher text. Tables 5-5 and 5-6 give the pairs of octets that are combined for the encryption or decryption operation for the example cases of n = 64 and n = 128. Table 5-5 is for the first LDU and table 5-6 is for the second LDU. The tables also give each octet to be encrypted a number, #, which is the Plain Text octet number in a superframe. # ranges from 1 to 112 for LDU1 and 113 to 213 for LDU2.

Table 5-5 Encryption Schedule for LDU 1, for n = 64 and 128

information		output n=64		output n=128		information		output n=64		output n=128		information		output n=64		output n=128	
F/O	#	B#	m#	B#	m#	frame/O	#	B#	m#	B#	m#	F/O	#	B#	m#	B#	m#
reserved	1	2 0	2 0			3 w7	41	7 0	4 8			7 w3	81	12 0	7 0		
reserved	2	2 1	2 1			3 w8	42	7 1	4 9			7 w4	82	12 1	7 1		
reserved	3	2 2	2 2			3 w9	43	7 2	4 10			7 w5	83	12 2	7 2		
L1	4	2 3	2 3			3 w10	44	7 3	4 11			7 w6	84	12 3	7 3		
L2	5	2 4	2 4			4 w0	45	7 4	4 12			7 w7	85	12 4	7 4		
L3	6	2 5	2 5			4 w1	46	7 5	4 13			7 w8	86	12 5	7 5		
L4	7	2 6	2 6			4 w2	47	7 6	4 14			7 w9	87	12 6	7 6		
L5	8	2 7	2 7			4 w3	48	7 7	4 15			7 w10	88	12 7	7 7		
L6	9	3 0	2 8			4 w4	49	8 0	5 0			8 w0	89	13 0	7 8		
L7	10	3 1	2 9			4 w5	50	8 1	5 1			8 w1	90	13 1	7 9		
L8	11	3 2	2 10			4 w6	51	8 2	5 2			8 w2	91	13 2	7 10		
1 w0	12	3 3	2 11			4 w7	52	8 3	5 3			8 w3	92	13 3	7 11		
1 w1	13	3 4	2 12			4 w8	53	8 4	5 4			8 w4	93	13 4	7 12		
1 w2	14	3 5	2 13			4 w9	54	8 5	5 5			8 w5	94	13 5	7 13		
1 w3	15	3 6	2 14			4 w10	55	8 6	5 6			8 w6	95	13 6	7 14		
1 w4	16	3 7	2 15			5 w0	56	8 7	5 7			8 w7	96	13 7	7 15		
1 w5	17	4 0	3 0			5 w1	57	9 0	5 8			8 w8	97	14 0	8 0		
1 w6	18	4 1	3 1			5 w2	58	9 1	5 9			8 w9	98	14 1	8 1		
1 w7	19	4 2	3 2			5 w3	59	9 2	5 10			8 w10	99	14 2	8 2		
1 w8	20	4 3	3 3			5 w4	60	9 3	5 11			LSD1	100	14 3	8 3		
1 w9	21	4 4	3 4			5 w5	61	9 4	5 12			LSD2	101	14 4	8 4		
1 w10	22	4 5	3 5			5 w6	62	9 5	5 13			9 w0	102	14 5	8 5		
2 w0	23	4 6	3 6			5 w7	63	9 6	5 14			9 w1	103	14 6	8 6		
2 w1	24	4 7	3 7			5 w8	64	9 7	5 15			9 w2	104	14 7	8 7		
2 w2	25	5 0	3 8			5 w9	65	10 0	6 0			9 w3	105	15 0	8 8		
2 w3	26	5 1	3 9			5 w10	66	10 1	6 1			9 w4	106	15 1	8 9		
2 w4	27	5 2	3 10			6 w0	67	10 2	6 2			9 w5	107	15 2	8 10		
2 w5	28	5 3	3 11			6 w1	68	10 3	6 3			9 w6	108	15 3	8 11		
2 w6	29	5 4	3 12			6 w2	69	10 4	6 4			9 w7	109	15 4	8 12		
2 w7	30	5 5	3 13			6 w3	70	10 5	6 5			9 w8	110	15 5	8 13		
2 w8	31	5 6	3 14			6 w4	71	10 6	6 6			9 w9	111	15 6	8 14		
2 w9	32	5 7	3 15			6 w5	72	10 7	6 7			9 w10	112	15 7	8 15		
2 w10	33	6 0	4 0			6 w6	73	11 0	6 8								
3 w0	34	6 1	4 1			6 w7	74	11 1	6 9								
3 w1	35	6 2	4 2			6 w8	75	11 2	6 10								
3 w2	36	6 3	4 3			6 w9	76	11 3	6 11								
3 w3	37	6 4	4 4			6 w10	77	11 4	6 12								
3 w4	38	6 5	4 5			7 w0	78	11 5	6 13								
3 w5	39	6 6	4 6			7 w1	79	11 6	6 14								
3 w6	40	6 7	4 7			7 w2	80	11 7	6 15								

F = Voice Frame # in a Superframe

O = Plain Text Octet description

= Plain Text Octet Number

B# = Encryption Block Number

m# = Number of Octet in B#

Table 5-6 Encryption Schedule for LDU 2, for n = 64 and 128

information		output n=64		output n=128		information		output n=64		output n=128		information		output n=64		output n=128	
F/O	#	B#	m#	B#	m#	frame	octet	#	B#	m#	B#	m#	frame	octet	#	B#	m#
10 w0	113	16	0	9	0	13 w7	153	21	0	11	8	17 w3	193	26	0	14	0
10 w1	114	16	1	9	1	13 w8	154	21	1	11	9	17 w4	194	26	1	14	1
10 w2	115	16	2	9	2	13 w9	155	21	2	11	10	17 w5	195	26	2	14	2
10 w3	116	16	3	9	3	13 w10	156	21	3	11	11	17 w6	196	26	3	14	3
10 w4	117	16	4	9	4	14 w0	157	21	4	11	12	17 w7	197	26	4	14	4
10 w5	118	16	5	9	5	14 w1	158	21	5	11	13	17 w8	198	26	5	14	5
10 w6	119	16	6	9	6	14 w2	159	21	6	11	14	17 w9	199	26	6	14	6
10 w7	120	16	7	9	7	14 w3	160	21	7	11	15	17 w10	200	26	7	14	7
10 w8	121	17	0	9	8	14 w4	161	22	0	12	0	LSD3	201	27	0	14	8
10 w9	122	17	1	9	9	14 w5	162	22	1	12	1	LSD4	202	27	1	14	9
10 w10	123	17	2	9	10	14 w6	163	22	2	12	2	18 w0	203	27	2	14	10
11 w0	124	17	3	9	11	14 w7	164	22	3	12	3	18 w1	204	27	3	14	11
11 w1	125	17	4	9	12	14 w8	165	22	4	12	4	18 w2	205	27	4	14	12
11 w2	126	17	5	9	13	14 w9	166	22	5	12	5	18 w3	206	27	5	14	13
11 w3	127	17	6	9	14	14 w10	167	22	6	12	6	18 w4	207	27	6	14	14
11 w4	128	17	7	9	15	15 w0	168	22	7	12	7	18 w5	208	27	7	14	15
11 w5	129	18	0	10	0	15 w1	169	23	0	12	8	18 w6	209	28	0	15	0
11 w6	130	18	1	10	1	15 w2	170	23	1	12	9	18 w7	210	28	1	15	1
11 w7	131	18	2	10	2	15 w3	171	23	2	12	10	18 w8	211	28	2	15	2
11 w8	132	18	3	10	3	15 w4	172	23	3	12	11	18 w9	212	28	3	15	3
11 w9	133	18	4	10	4	15 w5	173	23	4	12	12	18 w10	213	28	4	15	4
11 w10	134	18	5	10	5	15 w6	174	23	5	12	13						
12 w0	135	18	6	10	6	15 w7	175	23	6	12	14						
12 w1	136	18	7	10	7	15 w8	176	23	7	12	15						
12 w2	137	19	0	10	8	15 w9	177	24	0	13	0						
12 w3	138	19	1	10	9	15 w10	178	24	1	13	1						
12 w4	139	19	2	10	10	16 w0	179	24	2	13	2						
12 w5	140	19	3	10	11	16 w1	180	24	3	13	3						
12 w6	141	19	4	10	12	16 w2	181	24	4	13	4						
12 w7	142	19	5	10	13	16 w3	182	24	5	13	5						
12 w8	143	19	6	10	14	16 w4	183	24	6	13	6						
12 w9	144	19	7	10	15	16 w5	184	24	7	13	7						
12 w10	145	20	0	11	0	16 w6	185	25	0	13	8						
13 w0	146	20	1	11	1	16 w7	186	25	1	13	9						
13 w1	147	20	2	11	2	16 w8	187	25	2	13	10						
13 w2	148	20	3	11	3	16 w9	188	25	3	13	11						
13 w3	149	20	4	11	4	16 w10	189	25	4	13	12						
13 w4	150	20	5	11	5	17 w0	190	25	5	13	13						
13 w5	151	20	6	11	6	17 w1	191	25	6	13	14						
13 w6	152	20	7	11	7	17 w2	192	25	7	13	15						

F = Voice Frame # in a Superframe

O = Plain Text Octet description

= Plain Text Octet Number

B# = Encryption Block Number

m# = Number of Octet in B#

6. DATA OPERATION

Data messages (see Ref. 3, *Project 25 FMDA Common Air Interface*) are transferred over the CAI with a packet technique. Figure 6-1 shows how the message is broken into packets. The data message is first split into fragments. Additional information called auxiliary headers may be added to each fragment. Pad octets may be added so that each packet is an integral number of blocks. The message fragments are then formed into packets, consisting of a sequence of information blocks. Each block is protected by a trellis code, and the sequence of blocks is transferred over the common air interface as a single packet. Part of the packet is a check sum (CRC) to be used to verify the accuracy of the error correction in the receiver. Packets may be confirmed or unconfirmed. A confirmed packet adds an additional checksum to each block that allows corrupted blocks to be identified and resent using an automatic retransmission request (ARQ). The receiver then reassembles the packets into a continuous message.

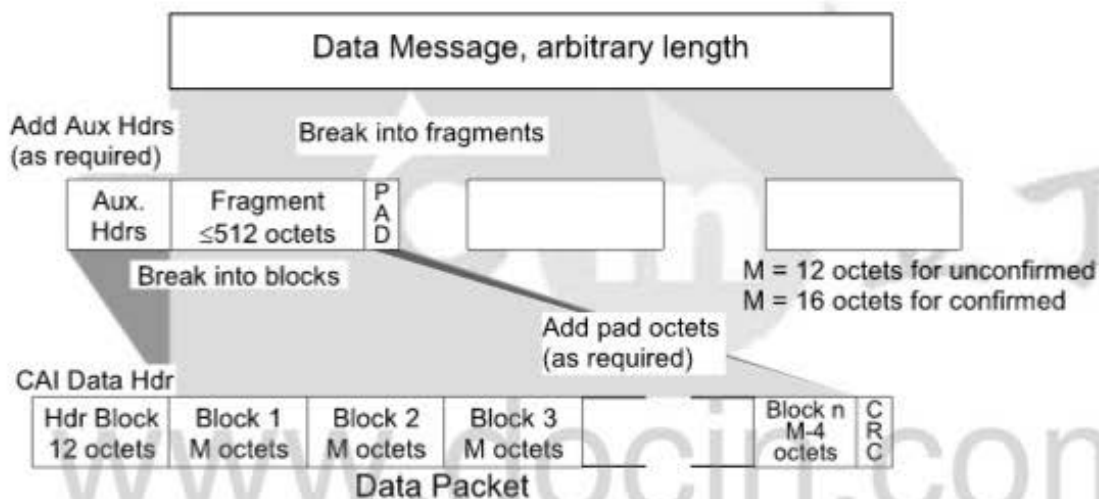


Figure 6-1 Decomposition of Data Message into Packets

Auxiliary Headers are treated as part of the user data. This means that the checksum (CRC) for the fragment includes the auxiliary header information. Auxiliary Headers may be chained to convey different types of information. In this chaining process, the CAI Data Header SAP would describe the first Auxiliary Header, the first Auxiliary Header would have a 2ndary SAP describing the 2nd Auxiliary Header, and so on until the last Auxiliary Header would describe the user data. Unlike the CAI data header, Auxiliary Headers may cross block boundaries.

Data is encrypted on a packet-by-packet basis. To enable decryption, an auxiliary header with Encryption Synchronization (ES) is added to each fragment. Any information, including Auxiliary Headers, after the ES is encrypted (except for the CRC and the pad octets that may or may not be encrypted). Any headers before the ES are not encrypted.

6.1 Encrypted Data Packet Structure

This ES Auxiliary Header provides the following information: a Message Indicator (MI), an Algorithm ID (ALGID), a Key ID (KID), and a secondary SAP. This amounts to 13 octets of information. The structure of the ES Auxiliary Header is shown in Figure 6-2.

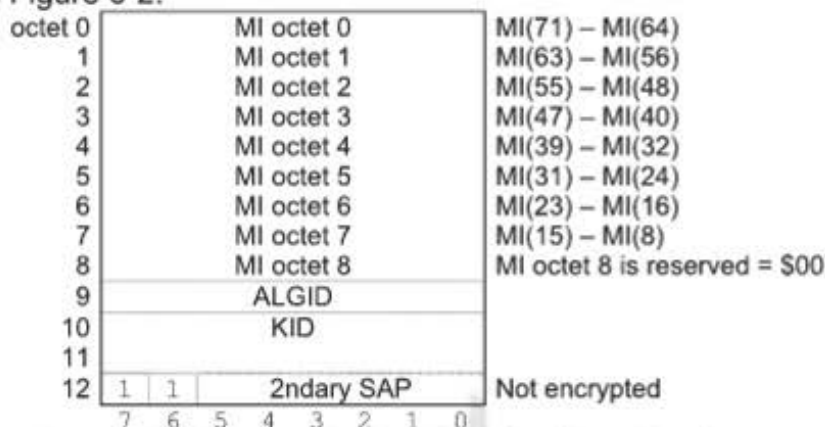


Figure 6-2 Encryption Synchronization (ES) Auxiliary Header

This figure shows the partitions for the MI, ALGID, KID, and SAP information. The MI octets 0 through 7 correspond to the left through right bits of the LFSR MI generator shown in figure 3-1. MI octet 0 consists of MI(71) through MI(64) and MI octet 7 consists of MI(15) through MI(8). MI octet 8 is used for other crypto implementations requiring a longer MI. For this Block Encryption Protocol, MI octet 8 is reserved and set to \$00.

The secondary SAP is used to designate the client for the next Auxiliary Header (if there is one) or the client above layer 2 of the plain text data (if user data follows the ES). If the ES is the only Auxiliary Header, then the secondary SAP operates in exactly the same way that the SAP in the header block would operate for unencrypted data. Hence, the encryption process transfers the SAP that would appear in the header block on unencrypted messages into the secondary SAP. To allow interpretation of the secondary SAP at intermediate points in the network, it is not encrypted. If there other Auxiliary Headers in addition to the ES, then the chaining rules given in section 6 apply.

Figures 6-3 and 6-4 show examples of the encrypted data packet structures for unconfirmed and confirmed modes for the case of the ES being the only Auxiliary Header. The Data Header Block is depicted on the first few lines. Details are given in reference 3, *Project 25 FDMA CAI*. An encrypted data packet is signified in the Header Block by a special Service Access Point identifier (SAP) value that is in octet 1 of the header. The Data Header Block also contains the Manufacturer's ID (MFID) that can be used for non-standard algorithms.

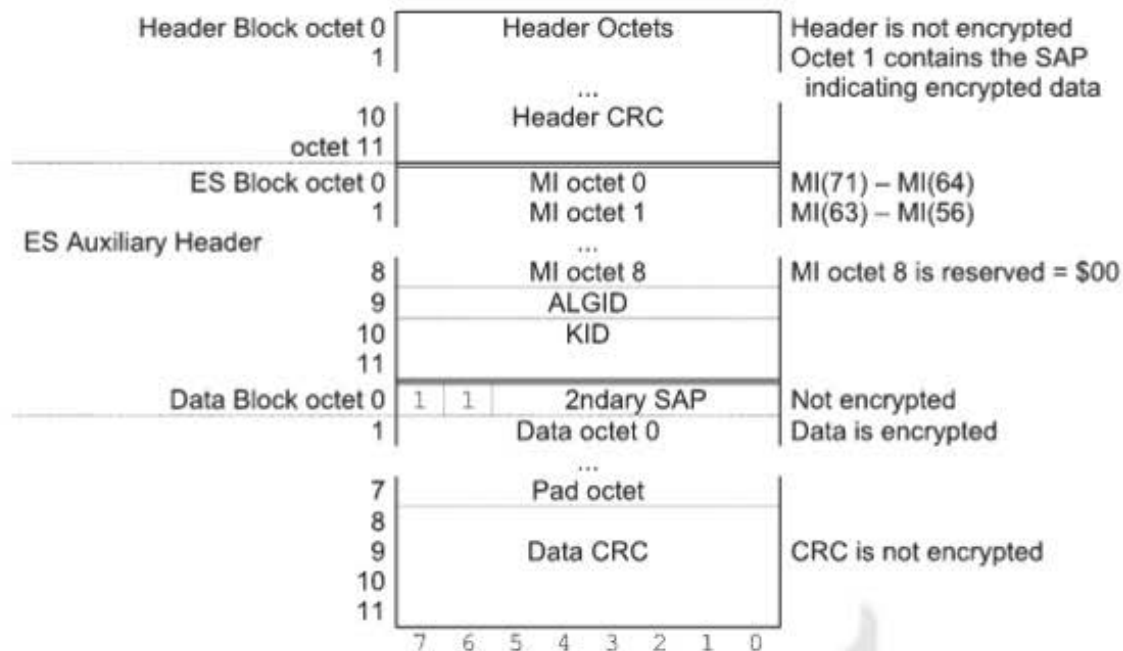
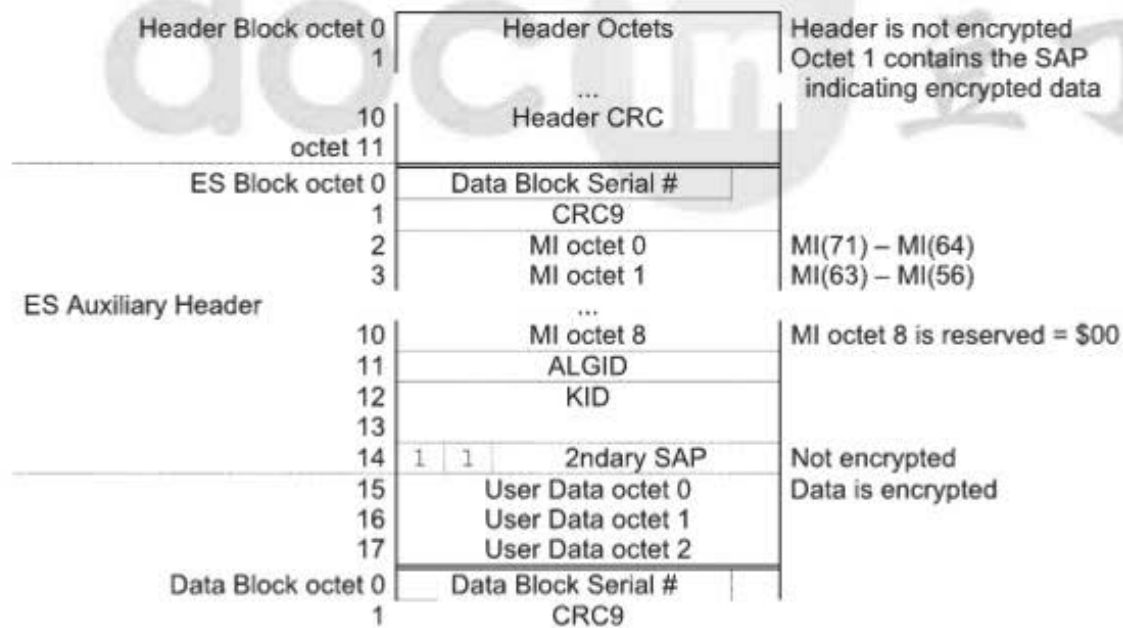


Figure 6-3 Encrypted Data Packet Structure Example, Unconfirmed



The encrypted data octets are in subsequent blocks of the packet. The number of blocks in the packet is given in the Header Block. Following the data octets are enough pad octets to extend the message to fill up the last data block. Following the pad octets are 4 octets for the data CRC as described in reference 3.

The data octets are encrypted using OFB exactly as described in section 4. The MI is loaded into the input register and encrypted. This encrypted result is not used for encryption, but is fed back to the input register to initialize the remaining OFB steps. The pad octets may or may not be encrypted. The data CRC is computed over the portion of the message starting with the first data octet following the Data Header Block and ending with the last pad octet (or data octet) in octet 7 of the last block for unconfirmed, or octet 13 of the last block for confirmed, as described in reference 3. For confirmed data, the Serial # and CRC9 octets are not included in the message CRC. Encrypted data is used in the computation of the CRC.

The data octets are encrypted by exclusive-ORing them with the output register value of the block algorithm. The output register octet numbers are shown in figure 5-2. Data octet 0 is exclusive-ORed with output register octet 0, data octet 1 is exclusive-ORed with output register octet 1, etc. After every m octets of data, there is another iteration of the OFB algorithm to yield another output register value for the next m octets of data. This process is continued for the duration of the packet. For example, if the packet contains 512 octets, the number of block encryption iterations is $(512/m)$ plus the first one that encrypts the MI.

www.docin.com

7. MANDATORY ALGORITHM

It is mandatory that the DES algorithm specified in Annex A, or a compatible algorithm be included in all equipment that implements this standard. A DES compatible algorithm is the TDEA algorithm using three identical keys as described in Annex B.



ANNEX A (NORMATIVE) – DATA ENCRYPTION STANDARD (DES)

A.1 Algorithm ID and Description

An Algorithm ID of \$81 (see reference 4, *Project 25 CAI Reserved Values*) shall indicate an encrypted message using the DES encryption algorithm. DES is a 64 bit block algorithm ($n = 64$) with a 64 bit key variable ($k = 64$, but every 8th bit going from left to right is parity). A complete description of this algorithm is given in

NIST, *Data Encryption Standard, FIPS Publication 46-3*, or
ANSI, *Data Encryption Algorithm, ANSI X3.92-1981*

A.2 Output FeedBack (OFB) Description

64 bit Output Feed Back is used. A complete description of this operation is given in

NIST, *DES Modes of Operation, FIPS Publication 81*, or
ANSI, *Data Encryption Algorithm – Modes of Operation, ANSI X3.106-1983*.

A.3 Bit Correspondence

In the DES description, bit 1 is the leftmost bit of the input/output register and bit 64 is the rightmost. In the terms of this protocol, bit 63 is DES bit 1 and bit 0 is DES bit 64.

www.docin.com

A.4 Voice Encryption Example

Given an MI = 1234 5678 90AB CDEF 00 (the first value given in Table 3-1), no expansion is required since $n = 64$. Then

DES Input Register = 1234 5678 90AB CDEF

For $n = 64$, $B = 28$ blocks are required to encrypt a voice superframe. $B\# = 1$ is not used for encrypting Plain Text.

For Key Variable = 0123 4567 89AB CDEF,
the following keystream shall be produced:

B#	Output Register	
1	BD66 1569 AE87 4E25	Not used for encrypting Plain Text
	Output Register = Keystream	
2	5D97 6A50 4786 581F	
3	5B02 29C3 4436 94E3	
4	78F8 7A8D 6DA5 72A3	
5	637B 8945 0941 03AB	
6	53AC E61C A2B1 9F5B	
7	2B46 85DE 9E98 4DC7	
8	97DF 0F1B 0D47 F725	
9	103A 5BE9 419D 7AB2	
10	F44E BE36 D9BE 5A9E	
11	4216 1755 15D8 1618	
12	3184 3C77 48EA E3FB	
13	84F6 E769 3B2D FD01	
14	4A8F 6D8A 2739 3EDB	
15	A50E 6ED9 1A1C 6AD5	
16	B3FC 7F75 D153 F97E	
17	B2F4 5199 38EC 6257	
18	CDA6 C0C7 5DAE 76B5	
19	8499 A7E7 0E83 FCCE	
20	9609 3257 7B2F 5810	
21	4AB0 4679 191E C5D4	
22	CFCD B169 4338 16CD	
23	D107 120B 6DFF CF72	
24	50C7 EEAC D9A3 3D7A	
25	ACD1 2FD9 AD8E 11B5	
26	4907 55A5 E022 E3BD	
27	6E9E AF38 DF82 7271	
28	8518 CD1B 78FF 57EA	3 rightmost octets not used

Given the following Plain Text for one voice superframe:

Plain Text	Link Control (LC) Information
00 00 00 00 01 00 00 01	
0C EE 42 22 6F 61 01 AB 31 EF 14	Voice Frame #1
0C EF 40 00 78 A7 01 A8 FD A9 AD	Voice Frame #2
10 EF 40 20 E8 A5 01 AC 55 16 EC	Voice Frame #3
18 F6 82 12 A8 46 01 BC F7 2A 1F	Voice Frame #4
80 DD 87 B1 8C 80 00 02 09 7B BA	Voice Frame #5
34 F0 AC 16 F0 4C 00 B3 BE 61 E5	Voice Frame #6
34 E9 74 60 B7 03 00 22 F0 69 3A	Voice Frame #7
30 DE 25 C4 08 61 00 67 AE 87 0B	Voice Frame #8
00 00	Low Speed Data
3C D2 86 DE 30 A4 00 08 D7 A2 B4	Voice Frame #9
04 E3 43 E6 F6 1F 02 E0 C1 8B AB	Voice Frame #10
0C EB 08 E9 94 5B 02 52 2A 9A D0	Voice Frame #11
A0 FC 5D 90 95 2F 88 83 2A 3C D3	Voice Frame #12
8C D9 1C A6 71 ED 00 01 09 FC 56	Voice Frame #13
98 DA 2B A2 87 C4 00 00 CE C9 6D	Voice Frame #14
30 DC 36 C0 DC 9D 00 49 B1 91 5A	Voice Frame #15
0C CF C1 41 12 9F 05 52 E6 E7 75	Voice Frame #16
0C EB 48 39 E5 26 00 2B 4D DD DC	Voice Frame #17
00 00	Low Speed Data
0C EE 02 E6 89 DD 00 B3 AA F1 85	Voice Frame #18

The following Cipher Text shall be produced:

Cipher Text	LC not encrypted (optional)
00 00 00 00 01 00 00 01	
CF AA 74 B6 8C 19 F9 D1 BC 82 B1	
7E 4C 23 7B F1 E2 08 E9 FE 02 FE	
BC 09 5C 82 59 3A 5A 87 13 93 32	
86 6E CF D5 3F 99 0E A7 FA 6D E8	
A5 CD BD EA 65 C1 9D 78 BB 8F F4	
8A C6 75 A8 AA D2 42 A5 A9 34 F0	
EC FF 6C 51 33 3F 77 6A 1A 8A C1	
B4 28 C2 AD 33 4C FD 66 E4 08 66	
8A 27	
05 EC 5D 7B 3E CA D9 12 CB C8 61	
B7 1F 3C 93 27 4C FB 9E 73 7F FA	
95 D3 E4 8B C3 96 A4 92 ED C7 7E	
D6 49 D9 09 32 C8 86 00 D6 F2 45	
85 EB 4B DD 5E B5 10 4B B9 BA 2F	
81 C4 EE 76 48 09 B1 69 8D F1 7B	
FD 0D 31 D2 D7 F0 FF 86 C3 C1 9D	
E2 63 18 E2 2F E5 A9 83 C9 3E D8	
82 FA FD 70 E2 73 A5 CB 6F 3E 61	
6E 9E	
A3 D6 DD 64 FB AC 85 AB 67 EA FD	

[This page left blank]



ANNEX B (NORMATIVE) – TRIPLE DATA ENCRYPTION ALGORITHM (TDEA), also known as Triple Data Encryption Standard (Triple DES)

B.1 Algorithm ID and Description

Encrypted messages using the Triple Data Encryption Algorithm (TDEA) are indicated by an Algorithm ID of \$83 (see reference 4, *Project 25 CAI Reserved Values*). TDEA uses 3 chained DES (see Annex A) encryptions/decryptions with a key bundle consisting of separate 64-bit DES key variables (K1, K2, and K3) for each encryption/decryption. The basic encryption operation can be described as

$$\text{Output} = E_{K3}(D_{K2}(E_{K1}(\text{Input})))$$

where E stands for a DES encryption operation and D stands for a DES decryption operation. Since the DES operations are chained, the size of the input and output registers is identical to DES, $n = 64$. A complete description of this algorithm is given in

ANSI, *Triple Data Encryption Algorithm Modes of Operation*, ANSI X9.52 - 1998.

An Algorithm ID of \$83 indicates the three-key version of TDEA (keying option 1 in the above standard). There are three independent 64-bit key variables, K1, K2, and K3 ($k = 192$, but every 8th bit going from the left to the right is a parity bit).

If $K1 = K2 = K3$, then TDEA is fully compatible with DES using K1, and the Algorithm ID shall be set to \$81 for compatibility with DES. Conversely, for a key variable with an Algorithm ID of \$81, a system using TDEA shall set $K3 = K2 = K1$ for compatibility with DES. (Of course, the key variable indicated by the Key ID must be held in common between the DES and TDEA systems).

B.2 Output FeedBack (OFB) Description

The TOFB64 mode of operation shall be used as specified the above document in the section "TDEA Output Feedback Mode of Operation."

B.3 Bit Correspondence

In the TDEA description, bit 1 is the leftmost bit of the input/output register and bit 64 is the rightmost. In the terms of this protocol, bit 63 is TDEA bit 1 and bit 0 is TDEA bit 64.

B.4 Voice Encryption Examples

Given an MI = 1234 5678 90AB CDEF 00 (the first value given in Table 3-1), no expansion is required since $n = 64$. Then

TDEA Input Register = 1234 5678 90AB CDEF

For $n = 64$, $B = 28$ blocks are required to encrypt a voice superframe. $B\# = 1$ is not used for encrypting Plain Text.

For Algorithm ID = \$83, with a Key Bundle of K1, K2, and K3, where

K1 = 0123 4567 89AB CDEF,

K2 = 2345 6789 ABCD EF01, and

K3 = 4567 89AB CDEF 0123,

the following keystream shall be produced:

B#	Output Register	
1	A011 B07C 7363 3375	Not used for encrypting Plain Text
	Output Register = Keystream	
2	F2EF 4174 6B0B EB27	
3	E18D F8D9 8D4A D4A9	
4	A9AF 375D A13B EEB0	
5	36F0 EAAD D77B 530D	
6	04A5 6B6F D803 5306	
7	1645 B1FB 8B19 686B	
8	6FB0 37C1 2A5F 1D8F	
9	B00B F92D FC4F 25C5	
10	5CE0 6D1B 14C4 8744	
11	A42A 6BEB 3A5C 827D	
12	8A8E CEF9 6BA1 6E6F	
13	F21A DA08 B048 F0C9	
14	16A8 4F68 DDFA 7AFF	
15	0EA9 4747 75A3 D6B0	
16	A2B8 865C 7506 C1CC	
17	0187 95DC DB57 B576	
18	9BC9 0859 894C 60DE	
19	EA0F 2BCC AC10 9C5D	
20	AFEF 557C 067C 1516	
21	3764 B87E 0AD2 4143	
22	C2AA 7FDB 19DA 3F04	
23	D167 6C36 B919 DBEE	
24	F6D9 AF8D 56B0 D527	
25	E769 D8D6 F5DC B5D7	
26	73FF 6C82 FC2D C063	
27	26F3 46A0 C2F0 A1F4	
28	1A21 4FC3 6F0A 87F8	3 rightmost octets not used

Given the Plain Text information for one voice superframe as shown in A.4, the following Cipher Text shall be produced:

Cipher Text

```
00 00 00 00 01 00 00 01
D5 63 08 F6 C6 C8 AE 9C 6C 4E 2F
E2 5F 76 F0 92 0A D6 D3 AE A4 A9
B5 84 2F F8 EB F6 07 BA 10 A7 17
93 EF EA 79 C7 F6 36 7D DD 75 02
0F 6D 8C 48 A1 7C 4F 27 CC 27 5A
59 EB B8 D2 77 08 A4 99 D5 8A DF
68 6B 09 EA 39 CD F9 49 51 07 55
C2 C4 FF CC B8 29 F0 AE B8 2F 44
68 DD
C6 A8 79 D0 99 E3 47 7D 74 74 04
```

LC not encrypted (optional)

```
A6 5B C5 BA 83 19 C3 2C C0 0C 3E
D0 30 5F 5C E2 C0 CB 5A 73 13 9C
C0 22 B7 9F BE E3 24 93 B6 61 7C
63 8C 60 A0 0D F8 16 36 6D 44 28
92 08 6A E1 45 6E 7F DB D7 13 52
34 0D 51 AC EA 24 19 92 5F 67 83
A3 42 97 F1 C7 B8 E2 3B 3E 31 80
D0 5E 9F 4A 1A 4A 82 D7 60 1D BF
26 F3
4A 4E C0 16 28 29 1A 92 E5 32 EA
```


[This page left blank]



ANNEX C (NORMATIVE) – ADVANCED ENCRYPTION STANDARD (AES)

C.1 Algorithm ID and Description

An Algorithm ID of \$84 (see reference 4, *Project 25 CAI Reserved Values*) shall indicate an encrypted message using the AES encryption algorithm with a key variable of 256 bits. A complete description of this algorithm is given in NIST, *Advanced Encryption Standard (AES)*, FIPS Publication 197, 26 November 2001, and is described by the notation AES-256. The following parameters shall be used:

Cipher Key Length = 256 bits ($N_k = 8$),
Input, Output, and state Block Length = 128 bits ($N_b = 4$), and
Number of Rounds = 14 ($N_r = 14$).

C.2 Output Feedback (OFB) Description

128-bit Output Feedback shall be used. In this mode of operation an input sequence (as described in section 4.2) shall be encrypted. The resulting output sequence is used as the next input sequence (the 0th output bit going to the 0th input bit and so on through to the 127th output bit to the 127th input bit). This output back to input is continued for 14 additional encryptions. The output sequence from each of these 14 encryptions is exclusive ORed with plain text to form the cipher text. This is also described in section 6.4 of NIST Special Publication 800-38A, Recommendation for the Block Cipher Modes of Operation, December 2001.

C.3 Bit Correspondence

In the AES description, bit 0 is the leftmost bit of the input/output and bit 127 is the rightmost bit of the input/output. In the terms of this protocol, bit 127 is AES-256 bit 0 and bit 0 is AES-256 bit 127.

C.4 Voice Encryption Example

Given an MI = 1234 5678 90AB CDEF 00 (the first value given in Table 3-1), expansion is required since $n = 128$. Then, from Table 4-1,

AES Input Register = 1234 5678 90AB CDEF 6FE2 802A A403 828B

For $n = 128$, $B = 15$ blocks are required to encrypt a voice superframe. $B\# = 1$ is not used for encrypting Plain Text.

For

Key Variable = 0123 4567 89AB CDEF 2345 6789 ABCD EF01 K(0) - K(127)
4567 89AB CDEF 0123 6789 ABCD EF01 2345 K(128) - K(255)

the following keystream shall be produced:

B#	Output Register	
1	A6E3 2E80 FE2E 0177 0E0F 9611 6943 CE93	Not used for encrypting Plain Text
	Output Register = Keystream	
2	A4A4 5220 6523 E33B AD35 8AF5 71CB 029A	
3	93AE FF9B DB7B 317F B85B 09F5 48D7 CE61	
4	9314 4CE0 8F62 A453 ABE5 E540 C10A 3E2F	
5	54F7 692D 707E BFDF E31C CF53 509C 890C	
6	30C3 3C9D D5D2 671C 98B5 7138 BF05 AE2F	
7	2250 D471 B0F2 C337 873B E6DB 55A5 12F3	
8	59C9 A851 7217 3EAF 102F 8AA4 48C5 D899	
9	B151 1BB4 E712 2841 0744 A389 DA84 5232	
10	D107 7133 3120 7577 848F 2626 A4AA 396A	
11	6FB9 AF20 90CF 197C EEE1 8570 D381 0895	
12	75F6 FA21 AA62 4780 25FE CEDA 2A8C 8F72	
13	F360 2C95 E8A6 BC8B 4509 6957 4CE8 F7B5	
14	321E 62FA B980 355B FA5E F396 BCC0 7E65	
15	B935 8601 7FB0 8621 4E4A 8B4C DAA7 DEB7	11 rightmost octets not used for encrypting Plain Text

Given the Plain Text information for one voice superframe as shown in A.4, the following Cipher Text shall be produced:

Cipher Text

```
00 00 00 00 01 00 00 01
F9 9F 89 20 F5 F2 AF 54 AA 34 6F
3D 90 F8 5B 71 52 49 7F 33 C8 3E
04 A3 A0 AF 8A 01 52 07 B0 F3 AC
D9 FC BC 3D FC B1 68 91 87 54 A0
5F 3E 9B 7E DF D0 9C 8B 05 4B 79
08 6D 79 C4 97 50 98 06 CF 59 5A
31 47 5B 42 E7 D7 71 92 02 AA 0D
B7 E5 C3 1F 5D C4 12 94 F7 4E A3
51 72
2B EC 29 CE 1F 2E A4 40 12 7A 2D
```

LC NOT ENCRYPTED (OPTIONAL)

```
B5 B2 58 52 11 0D 2A A1 C6 CF 08
85 31 8C BB A6 8A 05 23 19 AB F0
D5 8B D9 1F B3 09 2C 29 13 56 BC
35 76 3C 36 BE F4 7C EF E8 79 26
4B 5B 23 37 F2 32 FA 21 64 AB 2A
B0 F9 C8 0E 06 B7 8C C6 C3 62 3A
20 5A 29 E7 AE 14 40 5B 8F B0 39
E4 1C FD 0B FB 44 FA 92 CD E8 87
FA 5E
FF 78 BE 26 F7 B8 B9 86 2C F0 FA
```

[This page left blank]





